Task API + Webhooks — Guía completa (Docker + Postman + Newman + MongoDB)

Paso a paso para montar la API de tareas con Node.js/Express, MongoDB, Docker Compose y pruebas con Postman/Newman.

Requisitos

- Node.js 22.x + npm
- Docker Desktop / Docker Engine (WSL2 en Windows)
- Git
- Postman
- Newman (CLI) o imagen postman/newman

Estructura del proyecto

```
task-api-webhook/
 ■■ models/Task.js
 ■■ routes/tasks.js
 ■■ postman/
 ■   ■■ task-api-collection.json
 ■   ■■ task-api-env.json
 ■■ wait-for.sh
 ■■ server.js
 ■■ package.json
 ■■ .env
 ■■ Dockerfile
 ■■ docker-compose.yml
```

package.json (ESM)

```
{
  "name": "task-api-webhook",
  "version": "1.0.0",
  "type": "module",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "test:api": "newman run postman/task-api-collection.json -e postman/task-api-env.json -r cli,h
  },
  "dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "mongoose": "^8.5.0",
    "node-fetch": "^2.7.0"
  },
  "devDependencies": {
    "newman": "^6.2.1",
    "newman-reporter-htmlextra": "^1.23.1",
    "nodemon": "^3.1.4"
  }
}
```

.env

```
PORT=3000
MONGO_URL=mongodb://mongo:27017/taskdb
```

```
JWT_SECRET=supersecret
```

models/Task.js

```
import mongoose from "mongoose";

const TaskSchema = new mongoose.Schema(
  {
    title: { type: String, required: true, trim: true },
    description: { type: String, default: "" },
    completed: { type: Boolean, default: false },
    priority: { type: String, enum: ["Low", "Medium", "High"], default: "Medium" },
    dueDate: { type: Date }
  },
  { timestamps: true }
);

export default mongoose.model("Task", TaskSchema);
```

routes/tasks.js (incluye DELETE ALL)

```
import express from "express";
import Task from "../models/Task.js";

const router = express.Router();

router.get("/", async (_req, res) => {
  const tasks = await Task.find();
  res.json(tasks);
});

router.get("/:id", async (req, res) => {
  const task = await Task.findById(req.params.id);
  if (!task) return res.status(404).json({ message: "Task not found" });
  res.json(task);
});

router.post("/", async (req, res) => {
  const task = new Task(req.body);
  const saved = await task.save();
  res.status(201).json(saved);
});

router.put("/:id", async (req, res) => {
  const updated = await Task.findByIdAndUpdate(req.params.id, req.body, { new: true });
  if (!updated) return res.status(404).json({ message: "Task not found" });
  res.json(updated);
});

router.delete("/:id", async (req, res) => {
  const del = await Task.findByIdAndDelete(req.params.id);
  if (!del) return res.status(404).json({ message: "Task not found" });
  res.json({ message: "Task deleted" });
});

router.delete("/", async (_req, res) => {
  const r = await Task.deleteMany({});
  res.json({ message: "All tasks deleted", deletedCount: r.deletedCount });
});

export default router;
```

server.js (seed de 50 tareas si vacío)

```
import express from "express";
import mongoose from "mongoose";
import dotenv from "dotenv";
import cors from "cors";
import bodyParser from "body-parser";
import fetch from "node-fetch";
import taskRoutes from "./routes/tasks.js";
import Task from "./models/Task.js";

dotenv.config();
const app = express();
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(bodyParser.json());

let webhookUrl = null;

app.post("/webhook", (req, res) => {
  const { url } = req.body;
  if (!url) return res.status(400).json({ message: "Se requiere una URL" });
  webhookUrl = url;
  res.json({ message: "Webhook registrado", url });
});

app.use((req, _res, next) => { req.webhookUrl = webhookUrl; next(); });
app.use("/tasks", taskRoutes);

mongoose.connect(process.env.MONGO_URL, { dbName: "taskdb" }).then(async () => {
  console.log("■ Conectado a MongoDB");
  const count = await Task.countDocuments();
  if (count === 0) {
    const seed = [];
    for (i=1;i<=50;i++){ seed.append({title:f"Seed task {i}", description:"Initial load", complete
    # (Nota: en tu código real, reemplaza por las 50 tareas realistas que ya definimos)
    await Task.insertMany(seed);
    console.log("■ Inserted 50 tasks");
  }
  app.listen(PORT, () => console.log(`■ API escuchando en http://localhost:${PORT}`));
}).catch(err => console.error("■ Error de conexión:", err));
```

wait-for.sh

```
#!/usr/bin/env bash
set -e
hostport="$1"
shift
cmd="$@"
echo "■ Esperando a ${hostport}..."
until nc -z -v -w30 ${hostport%:*} ${hostport#*:}; do
  echo "Mongo no disponible aún... reintentando"
  sleep 1
done
echo "■ ${hostport} disponible, ejecutando: $cmd"
exec $cmd
```

Dockerfile

```
FROM node:22-alpine
WORKDIR /app
RUN apk add --no-cache bash curl netcat-openbsd
COPY package*.json ./
RUN npm install
COPY . .
```

```
RUN chmod +x /app/wait-for.sh
EXPOSE 3000
CMD ["/app/wait-for.sh", "mongo:27017", "node", "server.js"]
```

docker-compose.yml

```
services:
  api:
    build: .
    container_name: task-api
    ports:
      - "3000:3000"
    environment:
      - MONGO_URL=mongodb://mongo:27017/taskdb
      - JWT_SECRET=supersecret
    depends_on:
      - mongo

  mongo:
    image: mongo:6.0
    container_name: mongo-db
    restart: always
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

Levantar y probar

```
docker-compose up --build
curl http://localhost:3000/tasks
```